~~{Siemens AG}~~ **[Substitute Specification:]**

~~{New PCT application~~

~~26965-0726 (P-00,1968)~~

~~1998 P 02216 WO US~~

~~Inventor: Haas~~

~~Translation / January 24, 2001 / 1696(911) / 2040 words~~

~~CIRCUIT ARRANGEMENT}~~**[- - SYSTEM] AND METHOD FOR [ ]DATA CONVERSION IN A PROCESSOR**

[BACKGROUND OF THE INVENTION

Field of the Invention

The present invention belongs to the field of data processors. In particular, the present invention pertains to conversion of data in microprocessors.

Discussion of the Related Art]

In processor or microprocessors, ~~{a}~~ **[the terms will be used interchangeably herein,]** data conversion is produced by program sequences before arithmetic or logical operations **[are carried out]** with different data types. Data type conversion with program sequences~~{, however,}~~ has the disadvantage that it reduces the processing speed of a processor or microprocessor. Further, this kind of data type conversion ~~{exhibits}~~ **[has]** the disadvantage that the bus system of

the processor is additionally loaded by the operation code required for the data type conversion.

In addition to ~~{the}~~ **[this]** data type conversion ~~{that has been mentioned}~~, object-oriented program languages are ~~{utilized in order}~~ **[used]** to solve specific problems. A data type conversion can also be achieved by an object-oriented command structure. ~~{The}~~ **[However,]** object-oriented command structure~~{, however, produces}~~ **[has]** the disadvantage that each command for each combination of data types must be deposited in the memory. An enlargement of the program code likewise results in a reduction of the processing speed.

[SUMMARY OF THE INVENTION

Accordingly, it is an object of the present invention to provide a reliable and efficient system and] ~~{The invention is based on the object of specifying a circuit arrangement and a}~~ method for data type conversion ~~{that avoids the aforementioned disadvantages.}~~**[.]**

~~{This object is achieved by the features of patent claim 1 and 8. The invention yields the advantage that a }~~**[It is another object of the present invention to provide a system and method for]** data conversion can be implemented without reducing the processing speed of the processor.

~~{The invention yields the advantage that}~~ **[It is a further object of the invention to provide a system and method for data conversion wherein]** data type conversions are automatically implemented.

~~{The invention yields the further advantage that the program code }~~**[It is an additional object of the invention to provide a system and method for data conversion wherein the program code used]** need not be enlarged.

~~{The invention yields the further advantage that}~~ **[It is yet another object of the invention to provide a system and method for data conversion wherein]** a data type-suited data type conversion is implemented with a data type-suited address calculation.

~~{Further characteristics are recited in the subclaims.}~~ [BRIEF DESCRIPTION OF THE DRAWING FIGURES]

~~{The circuit arrangement and the method can be seen from the following, more detailed explanation of an exemplary embodiment with reference to drawings. Shown are:~~

~~Figure 1 }~~**[Figure 1 shows]** the structure of an object address **[according to the present invention;]**~~{;}~~

Figure 2 **[shows]** the structure of a register **[according to the present invention;]**~~{;}~~

Figure 3 **[shows]** the structure of internal registers **[according to the present invention;]**~~{;}~~

Figure 4 **[shows]** an embodiment of a processor **[according to the present invention]**; and

Figure 5 **[shows]** the embodiment of the processor with an object-oriented data type conversion according to the ~~{invention.~~

}[present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS]

Figure 1 shows a division of an object address OA. This object address OA is divided into an area for the specification of the type T of object and a memory address SA belonging to the object. The memory address SA indicates the memory location of an object in a memory area of a memory. The object type T and the memory address SA are assigned by a compiler given a declaration of the object and are treated as a unit. For example, objects can be data, addresses or code addresses.

Figure 2 shows the division of a register R. This register R is divided into a first area for indicating the data or, respectively, object type T and into a second area for storing the data or, respectively, objects D.

Figure ~~{s}~~ **[3]** shows a detailed specification of the division of the register R and of the object address OA. In this illustration, the addresses or data to be stored are divided into a first and second area, as indicated above. For example, 3 bits for the indication of the object type T and 32 bits for the object to be stored, the address or data are provided in a register R having a length of 35 bits. The 32 bits for the object can be a data word, a data or code address or a memory address with a particular about the type of memory address. Further subdivisions are shown in Figure 3.

Figure 4 shows a processor, particularly an RISC **[(reduced instruction set code)]** processor. In this illustration, the critical components of the ~~{RSIC}~~ **[RISC]** processor arranged in a pipeline are a sequence controller SC, an instruction decode ID, a register read unit RR, an execute unit E, a data transfer unit DT and a

register write unit RW. Data are either read in or out from an external storage unit M via the data transfer unit DT. Data that are of significance for the ongoing processing process are deposited in a register file RF via the register write unit RW and are in turn read as needed by the data read unit RR. Logical and arithmetic instructions are executed in the execute unit E.

In Figure 5, the RISC processor is ~~{fashioned}~~ **[formed]** with at least one data type conversion unit TC that ~~{implements}~~ **[carries out]** a data type conversion. This data type conversion unit can~~{, for example,}~~ be arranged before and after the execute unit E or between the external storage unit M and the data transfer unit DT. The data type conversion unit can likewise be arranged between the register write unit RW and the register file RF and between the register file RF**[,]** and the register read unit RR in order to implement a data conversion that becomes necessary. ~~{The illustrated arrangement with}~~ **[As shown in Figure 5,]** a data type conversion unit TC in the data pipeline of the RISC processor ~~{yields the advantage}~~ that the clock frequency of the pipeline remains unmodified.

In the present invention, the object address OA is divided into a first and second area. The data type conversion unit T^C determines the data type T from the first area and the physical address SA from the second area.

Given a memory access, an address calculation of indicated load and store instructions ~~{[...]}~~ **[is made]** with the assistance of the object type~~{[...]}~~**[, before]** an address offset corresponding to the object size is determined for the physical address SA.

Given a load instruction, the memory address of the object is known. The object is deposited in a register of the processor with the indication of the type.

Given a store instruction, the data to be stored and the object type ~~{appertaining}~~ **[pertaining]** thereto are taken from the register, converted into the object type of the destination address and deposited under the destination address.

In addition to containing the register numbers of source and destination registers, execute instructions also contain the ~~{appertaining}~~ object type for the destination register. The object type of the operation is derived from the object type of the destination register. A processing unit in the execute unit E ~~{for, for example,}~~**[, such as]** an arithmetic operation**[,]** is selected on the basis of the object type. Since the object types of the source and destination registers are known before an arithmetic operation, a data type conversion can be ~~{correspondingly}~~ undertaken. Before an operation, the source data are converted into a data type that can be processed by an execute unit E of an RISC processor. After an operation, the result can be converted into the data type of a destination register.

Given a multiplication of a first and second variable, the compiler ~~{respectively}~~ generates the physical address and the data type. When loading the first and second variable, the data types are deposited in the register file RF. The program section – load variable 1 – signals the size of the first variable to the processor, for example an 8 bit value or 16 bit value. With the data type, the data type conversion unit TC is informed whether it is a matter of a signed integer variable**[,]** or of an unsigned integer variable. After the loading of the first variable, the data belonging to the second variable are loaded. Since, upon reading the data from the register file RF, the data type conversion unit TC knows the two data types of the first and second variable, the data types are adapted to one another and the

actual operation such as, for example, a multiplication is executed in the following execute unit E.

The data type conversion unit TC can be introduced into the data pipeline of the RISC processor at various locations following the execute unit E. A lack of the data type conversion unit TC following the execute unit E results ~~{therein}~~**[, such]** that the result data ~~{that were}~~ formed in the execute unit E are deposited in the register file RF together with a data type deriving from the respective operation.

Given a store instruction, both the data as well as the respective data type are again read from the register file RF and supplied to a data type conversion unit TC preceding the execute unit E. The data type conversion according to the data type of the destination address ~~{can ensue}~~ **[follows from]** there.

Given an indexed addressing, the data type conversion unit TC need only be informed of the simple index, and the processor can calculate the byte offset on the basis of the data type of the address. Given a normal integer value that exhibits a length of 2 bytes~~{,}~~ for example, it is thus known that the index must be ~~{correspondingly}~~ **[correspond-ingly]** doubled and added to the physical address ~~{therefor}~~. This ~~{yields}~~ **[has]** the advantage that time-consuming conversions for determining the address indices are eliminated.

~~{Another}~~ **[An]** advantage of the indexed address calculation is ~~{comprised therein}~~ **[based upon the fact]** that a table of data with short integer values ~~{or}~~**[,]** long integer values or normal integer values can be processed with one and the same program code. ~~{The}~~ **[Therefore, the]** program code ~~{thereby}~~ need not distinguish between ~~{said}~~ **[the]** data types**[,]** since the processor calculates its addresses itself on the basis of the data types.

~~{Patent Claims}~~ **[As an alternative to using a RISC processor, the present invention can be implemented using a CISC (complex instruction set code) processor in the same manner as described above for the RISC processor.]**

~~{1. Circuit arrangement for data conversion in a processor having at least one unit (E) executing a logical or arithmetic operation, characterized in that an object-oriented data conversion unit (TC) for recognizing a type (T) of an object (D) and an object address (OA) is arranged preceding the unit (E) executing the logical or arithmetic operation, and, based on the type information accompanying an object address (OA) and the object (D), the data conversion unit (TC) recognizes the type (T) of the object (D) and, given non-uniformity, matches the objects (D) before an operation or generates a predetermined type of object (D).~~

~~2. Circuit arrangement according to claim 1, characterized in that a memory location for an object address (OA) and a memory location of a register (R) is respectively divided into a first and second area (T, SA; T, D), whereby a type (T) of the object (SA, D) is respectively deposited in the first area.~~

~~3. Circuit arrangement according to claim 1, characterized in that the object-oriented data conversion unit (TC) is provided following the unit (E) executing a logical or an arithmetic operation.~~

~~4. Circuit arrangement according to claim 1, characterized in that the object-oriented data conversion unit (TC) is arranged preceding the storing of the object (D) in an external storage (M) and a register file (RC).~~

~~5. Circuit arrangement according to claim 1, characterized in that a register file (RC) is divided into a memory area for data and a memory area for a respective type indication of the data.~~

~~6. Circuit arrangement according to claim 1, characterized in that this is a reduced instruction set computer (RISC).~~

~~7. Circuit arrangement according to claim 1, characterized in that this is a complex instruction set computer (CISC).~~

~~8. Method for data conversion in a processor having at least one unit (E) executing a logical or arithmetic operation, characterized in that an object-oriented data conversion is implemented by a type information (T) in an object address (OA) and by a type information (T) of an object (D), and, given an inequality of the objects (D) to be operated by a logical or arithmetic operation, the type of the objects is matched to one another or a predetermined object type of an object (D) is generated.~~

~~9. Method according to claim 8, characterized in that a memory location for an object address (OA) and a memory location of a register (R) is respectively divided into a first and second area (T, SA; T, D), and a type information of the memory address (SA) deposited in the second area of the object address (OA) and the data (D) of the register (R) deposited in the second area is respectively noted in the first area (T).~~

~~Abstract~~

~~Circuit Arrangement and Method for Data Conversion in a Processor~~

~~A data conversion to be implemented in a processor is implemented object-oriented before an arithmetic or logical operation, being respectively implemented on the basis of a data type indication belonging to the object.~~

~~Figure 3~~ **[Although preferred embodiments of the invention have been described herein, it is to be understood that the invention is not limited to these embodiments, but that various changes and modifications thereto may be made by persons having skill in the art to which this invention pertains,**

without departing from the scope and spirit of the invention, which is to be defined by the following claims. - -]